1. Code Quality and Best Practices

1. Why?
   * 1. Best Practices help improve rendering of the website
     2. Code becomes more organised
     3. It gets easier to read and understand the code
2. Solution
   * 1. Use Lighthouse (A chrome extension) to get your website graded on basis of FOUR categories.   
        <https://developers.google.com/web/tools/lighthouse/>
     2. Use HTML5 Validator  
        <https://validator.w3.org/>
     3. Use CSS3 Validator  
        <https://jigsaw.w3.org/css-validator/>
     4. Use JavaScript and jQuery Validators
        1. <http://www.jslint.com/>
        2. <http://jshint.com/>
     5. Use HTTPS (CloudFlare)  
        <https://www.techromance.com/2017/08/02/simple-steps-convert-http-site-https/>
     6. Use Caching and a service worker

2. URL Format

1. Why to take care of URL format?
2. Avoid use of file extensions wherever possible.
3. File extensions appear at the end of web addresses, and have several negative effects.
4. They make the address harder to remember or type (particularly for non-technical users), and can reveal the underlying technology of the website making it very slightly more vulnerable to hackers.
5. They also tie the implementation of the website to a specific technology, which can make subsequent migration of URLs difficult.
6. Consider URL rewriting as an effective and transparent means of creating appropriate URLs
7. Solution –
   * 1. Use .htaccess file
        1. <http://www.htaccess-guide.com/>
        2. <https://alexcican.com/post/how-to-remove-php-html-htm-extensions-with-htaccess/>

3. Minify and compress (gzipping) JavaScript Files, CSS Files and images

1. Why?
   * 1. Code comments are removed.
     2. Line breaks are removed.
     3. Unnecessary spaces are removed.
     4. Extraneous punctuation (such as parentheses and semicolons) and other whitespace are removed.
     5. Some JavaScript minifiers do more advanced operations like shortening of variables, properties, arguments, classes, functions and method names (sometimes people call this to be obfuscation but is just a bi-product of minification)
     6. Some JavaScript minifiers wrap the code in a immediately executed function, with a lot of arguments like below that makes it possible to use these variables without declaring them with the var keyword, thus reducing the size of you code in some cases with three bytes times the number var keywords.
2. Solution
   * 1. Fastest and the most preferred method is to use GruntJS or GulpJS.  
        <https://www.youtube.com/playlist?list=PLvZkOAgBYrsRmqqKk6W1O4HKVrlaZsPAc>
     2. Manual compression of images can be done using tinyPNG  
        <https://tinypng.com/>

-------------------------------------------------------------------

Small Note:-

STEPS PERFORMED BY BROWSER

1. Process HTML mark-up and build the DOM tree.
2. Process CSS mark-up and build the CSSOM tree.
3. Combine the DOM and CSSOM into a render tree.
4. Run layout on the render tree to compute geometry of each node.
5. Paint the individual nodes to the screen.

***Optimizing the critical rendering path* is the process of minimizing the total amount of time spent performing steps 1 through 5 in the above sequence.** Doing so renders content to the screen as quickly as possible and also reduces the amount of time between screen updates after the initial render; that is, achieve higher refresh rates for interactive content

<https://developers.google.com/speed/docs/insights/v2/reference/>

-------------------------------------------------------------------